UNIVERSITÀ DI PISA

Università degli Studi di Pisa

Laurea Specialistica in Tecnologie Informatiche

Advanced Programming

Mid Term Paper

Start Date: 5/04/2017
Submission deadline: 15/04/2017 (send a single PDF file to attardi@di.unipi.it)

Rules:

The paper must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- **to consult or setup an online forum, to request help of consultants in producing the paper**
- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone"s code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

The paper must:

1. be in a single PDF file, formatted readably (**font size ≥ 10 pt** with suitable margins, single column), of **no more than 10 numbered pages**, including code: for each extra page one point will be subtracted from the score.
2. include the student name
3. provide the solution and the code for each exercise separately, referring to the code of other exercises if necessary.
4. cite references to literature or Web pages from where information was taken.

Introduction

We will develop a simplified blockchain mechanism like that of BitCoin (https://en.wikipedia.org/wiki/Bitcoin).
Each user of the system has an associated keypair (private, public), that uses to sign transactions.
A transaction consists of:

- list of inputs
- list of outputs
- signature

A transaction input consists of an UTXO (Unspent Transaction Output):

- hash256 identifier of a previous transaction
- index in list of output of that transaction

Outputs consists of:

- value
- public key of the beneficiary

Suppose Alice wants to send 40 BTC to Bob, spending money from two previous transactions:

- TX1: $inputs_1$, [<30, PK_A>]
- TX2: $inputs_2$, [<15, PK_A>]

where PK_A is the public key of Alice, and PK_B is the one by Bob. A new transaction TX4 can be made consisting of:

TX4: [<$Hash_{TX1}$, 0>, <$Hash_{TX2}$, 0>], [<40, PK_B>, <5, PK_A>], signature(TX4, SK_A)

where signature(TX4, SK_A) is the signature of TX4 (inputs and outputs) produced with the private key SK_A of Alice. Such a signature allows anyone, using the public key PK_A of Alice, to verify that the transaction was made by her.
Notice there is an output of 5 BTC going back to Alice for the rest of the amount.
This transaction will be broadcast to all BitCoin nodes for validation.
Network nodes can validate transactions, add them to their copy of the blockchain, a public ledger that records bitcoin transactions, and then broadcast these ledger additions to other nodes. Periodically a new block of accepted transactions, is created, added to the blockchain, and quickly published to all nodes.
A blockchain is a series of linked blocks, where each block contains:

1. The number of the block in the list.
2. The hash of the previous block in the chain.
3. A nonce value which is an integer (chosen so that the block's hash value starts with three 0's).
4. The transactions in the block.

Exercise 1

Design a set of classes to represent `Hash`, `Transaction` and `Block`.
The `Hash` class should provide a method `isValid()` to check that the hash values starts with three 0's, and a method `equals()`. (Hint. In Java you might use class `MessageDigest` for implementing cryptographic hashes).
For asymmetric cryptography, you can use RSA, for example in Java see this tutorial:
http://www.java2s.com/Tutorial/Java/0490__Security/AnRSAsampleapplication.htm.

Exercise 2

Define the class `BlockChain`, that represents a ledger and implements method `isValid()` that walks the blockchain and ensures that its blocks are consistent and valid (including that there are no double payments and each payment has appropriate coverage), and method `getBalance()` that computes the balance for a given user.

Exercise 3

Write a program that allows interactively manipulating a blockchain and mining for additional blocks. The program creates a blockchain with an initial dollar amount and then repeatedly:

1. Prints out the contents of the blockchain.
2. Reads in a command.
3. Executes that command, possibly updating the blockchain and reporting back to the user.

The program should support these commands:

```
mine: discovers the nonce for a given block
transfer: perform transaction
remove: removes the last block from the end of the chain
check: checks that the block chain is valid
report: reports the balance of a given user
quit: quits the program
```

The command check should verify that there are no double spending in the blockchain.

Assume there is a map that associates to user names their keypair with at least three users.

Show an example of execution where a chain is valid, then becomes invalid, then is restored to valid.

Exercise 4

Illustrate the concept of delegates in C# and explain the difference with respect to closures.